

Design and Fabrication Warehouse Robot Delivery Device with Machine Learning for Obstacle Avoidance

Lijo Jacob Varghese

*Department of Mechatronics Engineering,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
lijojacobvarghese@skcet.ac.in*

Parthasarathi HG

*Department of Mechatronics
Engineering,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
vasan1652009@gmail.com*

Pranav MV

*Department of Mechatronics
Engineering,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
mvpranav29@gmail.com*

Praneeth CS

*Department of Mechatronics Engineering,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
praneethshivakumar@gmail.com*

Sree Omkar PA

*Department of Mechatronics Engineering,
Sri Krishna College of Engineering and
Technology,
Coimbatore, India
sreeomkar007@gmail.com*

Abstract —Increased use of Industry 4.0 and smart logistics technologies has forced the warehouses to adopt autonomous transport mechanisms for material transport and minimizing the human effort and labor. This paper presents the development of a six-wheeled, all-terrain Warehouse Delivery Robot using six 12 V, 60 RPM Johnson geared DC motors designed to carry goods through a warehouse floor without encountering obstacles. An ESP32 microcontroller is employed as the main component of the embedded controller responsible for motor drive control, communication with a computer and sensor data integration. A Python application developed on the computer is used to command destination and movement of the robot via a Wi-Fi network. Obstacles are recognized using a combination of TensorFlow/YOLO object recognition algorithm (pre-processed using OpenCV library) that recognizes pallets, boxes, employees and other objects from images captured from the onboard cameras along with ultrasonic range detection mechanism which helps the robot to slow down, turn or change direction without stopping due to an obstacle ahead. Path planning is also done by an occupancy grid map of a SLAM style created using ultrasonic sweep and odometry which are then analyzed using A* shortest path search algorithm to find the shortest way from the robot to a desired location before taking over with the reactive part of the system. The bench and floor tests of the developed robot showed successful obstacle recognition, closed loop performance of motors and comfortable torque margin for 10 kg total weight of the robot and payload

Keywords — Warehouse Robot, ESP32, Machine Learning, Python, Autonomous Navigation, Obstacle Avoidance, Johnson DC Motor, Industry 4.0, Smart Logistics, YOLO, SLAM, A* Path Planning.

I. INTRODUCTION

The synergy of Industry 4.0, artificial intelligence and warehouse automation has changed the way goods are handled in modern facilities. Modern warehouses and distribution centres need efficient delivery of materials from storage racks to production lines and dispatch bays, but the majority of such operations rely on manual labor — a dependence which not only drives up operational costs but slows down material flow

and leaves room for errors and accidents. Autonomous mobile robots offer a safer and more reliable solution to that challenge.

This paper describes the Warehouse Delivery Robot with obstacle avoidance that is designed to carry goods autonomously inside the warehouse environment. This robotic vehicle features an all-terrain design based on a six-wheel chassis, driven by 12 V, 60 RPM Johnson geared DC motors and 100 mm wheels allowing steady operation over rough floors, inclines and uneven ground, and carrying moderate loads. All these features were taken into account while designing the mechanical part of the robot to ensure stable work in warehouse conditions.

The control system of this robot is built around ESP32 microcontroller responsible for motor control, reading the sensors' output and communication. An operator controls the robot via Python desktop application sending destination/movement requests which are then transmitted wirelessly to ESP32 over Wi-Fi connection. The navigation module relies on machine learning; the robot uses its camera to get live video feed and a Python-based algorithm recognizes such objects as boxes, pallets, people, forklifts and other warehouse items. Paired with ultrasonic distance readings, this lets the robot pick a safe path and keep moving toward its destination instead of stopping outright the moment an obstacle appears.

Perception and decision-making are kept separate from low-level actuation by design. The Python host carries the computational load of classification, while the ESP32 handles motor response on its own, with deterministic, low-latency timing. Splitting the work this way keeps the robot responsive even when the vision pipeline is briefly occupied — a detail that matters on a warehouse floor shared with people and equipment in motion.

The contributions of this work are fourfold: (i) a low-cost six-wheel drivetrain suited to uneven warehouse floors; (ii) a Python-ESP32 command architecture for operator-directed delivery; (iii) a vision- and ultrasonic-based obstacle-

avoidance pipeline capable of intelligent re-routing rather than simple stop-on-detection behaviour; (iv) a lightweight SLAM-based occupancy-grid map paired with an A* shortest-path search for autonomous route selection; and (v) an experimental evaluation of detection accuracy, response latency, path-planning performance, and torque margin on a fabricated prototype.

Cost is a further driver behind this design. Commercial automated guided vehicles (AGVs) typically lean on LiDAR-based simultaneous localisation and mapping (SLAM), which is precise but expensive and often more capability than small or medium warehouses need for straightforward point-to-point delivery along known aisles. This work instead builds its mapping and path-planning layer from the same camera and ultrasonic sensors used for obstacle detection, avoiding a dedicated LiDAR unit while still giving the robot a shortest-path SLAM-style capability, alongside the learning-based obstacle handling that sets it apart from simple bump-and-turn robots.

The remaining part of the paper is structured as follows. Section II presents the relevant work on embedded robot control, vision-based obstacle avoidance, design of drivetrain, and path planning. Section III provides the system architecture and implementation of the proposed approach that includes SLAM based shortest-path layer. Section IV describes the methodology used for experiments and results obtained. The prototype cost analysis is provided in Section V. Section VI discusses the results and limitations. Section VII highlights the future scope of the study.

II. RELATED WORK

Indoor and warehouse movement is currently gaining much interest in using low-cost microcontrollers together with visual sensors and ultrasonic sensor systems to achieve reliable obstacle detection, but without the cost of LiDAR technology. Research conducted on ESP32 microcontroller platforms indicates that the microcontroller is ideal for integrating both Wi-Fi communication and motor control into one low-powered module for use with material handling robots [1][2].

Complex algorithms such as YOLO working within the context of the OpenCV framework have been the other traditional method for going beyond the basic approach of avoiding the object through distance and making robots able to distinguish between the static and dynamic objects such as pallets and humans, respectively [3][4]. Another popular approach has been the ultrasonic ranging using HC SR04 sensors which has been popular because of its cost-effectiveness and short-range measurements.

On the mechanical side, geared-DC-motor drivetrains with multiple wheels have been studied for their traction and load-carrying ability on uneven or ramped indoor floors, with six-wheel layouts offering better stability than four-wheel ones under moderate payloads [6]. Closed-loop control through PWM-driven high-current drivers such as the BTS7960 has likewise been shown to give smoother speed regulation than simple on/off switching for geared DC motors [7].

Wi-Fi socket links between a desktop application and an embedded controller have appeared in several logistics-robot prototypes, letting an operator send destination commands without a dedicated remote and with lower latency than SMS/GSM-based schemes indoors [8][9]. Against Bluetooth, a Wi-Fi socket link also scales more naturally to multi-robot fleets on a shared warehouse access point, since each unit can be addressed on its own over the same network.

Studies on Li-ion battery packs regulated through buck converters guide the choice of a stable 5 V rail for powering an ESP32, camera, and sensor suite from a single 12 V pack [10]. Stable regulation matters here in particular, since voltage sag under the peak current of six motors starting together can otherwise reset a poorly decoupled microcontroller supply.

A detailed comparison table is left out for brevity, but broadly speaking, ultrasonic- or infrared-only robots are the cheapest option yet cannot tell obstacle types apart; LiDAR/SLAM-based AGVs give the richest spatial awareness at the highest cost; and vision-plus-ultrasonic hybrids of the sort proposed here sit in between, adding object-level understanding over plain ranging at a fraction of LiDAR's cost, with some added sensitivity to lighting.

Grid-based path planning using algorithms such as A* remains a standard approach for finding a shortest route once an occupancy map is available, valued for combining optimality guarantees with a modest computational cost suited to embedded or laptop-class hosts [16]. Lightweight SLAM techniques built from sonar or vision-based ranging rather than LiDAR have similarly been used to produce serviceable occupancy grids for indoor mobile robots at a fraction of LiDAR-SLAM's sensing cost, at some expense to map resolution and drift correction [17].

Taken together, this body of work points toward a layered design for the proposed robot: a camera with a Python-based ML pipeline for obstacle classification, ultrasonic ranging for distance cross-checking, a six-wheel Johnson-motor drivetrain with BTS7960 closed-loop control for locomotion, a lightweight SLAM-style occupancy grid with A* shortest-path search for route planning, and a Wi-Fi socket link between a Python operator GUI and the ESP32 for command and status reporting. A gap that recurs across the surveyed literature is the limited pairing of learning-based obstacle classification with multi-wheel, ramp-capable drivetrains built specifically for warehouse floors, together with an onboard, low-cost path-planning layer — a gap this work addresses directly through an integrated hardware–software prototype and its experimental evaluation.

III. METHODOLOGY AND IMPLEMENTATION

This work presents a modular Warehouse Delivery Robot architecture combining an operator-directed command layer, onboard visual and ultrasonic perception, a SLAM-based path-planning layer, a machine-learning navigation-decision layer, and closed-loop motor control. ML-based obstacle classification runs on a Python host, separate from the low-level embedded motor control on the ESP32, so the system can make intelligent routing decisions while still guaranteeing responsive, safe movement through dedicated hardware.

A. System Architecture

The perception layer pairs a camera with a Python-based machine-learning model (TensorFlow/YOLO with OpenCV pre-processing) that captures and analyses image frames to identify warehouse objects such as cartons, pallets, workers, and forklifts. An ultrasonic ranging module runs in parallel, continuously measuring distance to the nearest obstacle ahead of the robot, and its sweep readings together with wheel-motion data are also used to build and update a lightweight occupancy-grid map of the aisle for SLAM-based route planning. A navigation-decision engine draws on all three streams — classification result, measured distance, and the current shortest path from the A* search over the occupancy grid — to decide whether the robot should continue, slow

down, turn, or select an alternate route. This decision is sent to the ESP32 over the Wi-Fi socket link, which commands the motor driver accordingly, while the operator GUI runs in parallel, issuing the original destination command and receiving status updates back from the ESP32.

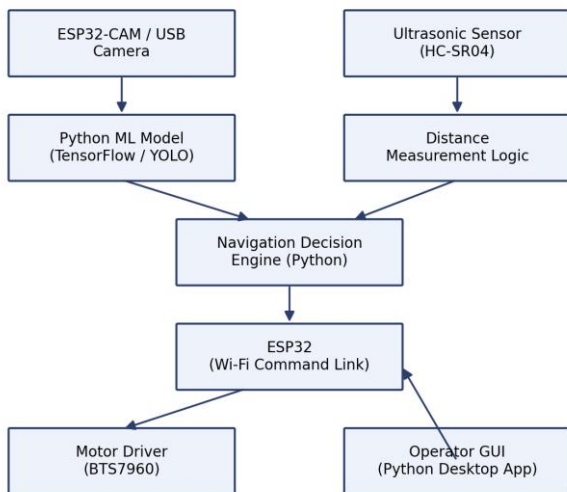


Fig. 1. System, AI/ML, and SLAM path-planning integration architecture.

B. Working Principle

An operator begins by entering a destination or movement command in the Python GUI, sent to the ESP32 over Wi-Fi. The ESP32 drives the six Johnson geared DC motors through the BTS7960 driver, moving the robot along the shortest path computed over the occupancy-grid map for its assigned destination. While moving, the ultrasonic sensor keeps checking for nearby obstacles and the camera streams frames to the Python ML model for classification. When an obstacle is detected, the navigation logic tells the ESP32 to slow down, turn, or recompute a short alternate route on the grid, after which the robot returns to its planned path toward the destination. When it reaches there, it halts and signals completion of the assigned task to the user application.

C. Hardware and Control Integration

Hardware layer links software components with the physical structure of the robot. An ESP32 development board acts as the central processing and communication unit, interfacing with the camera, the ultrasonic sensor, the Wi-Fi link to the Python application, the BTS7960 motor driver, and the six Johnson geared DC motors. A 12 V Li-ion battery pack, guarded by a BMS and regulated through an LM2596 buck converter, supplies steady power to all onboard electronics and sensing modules.

Each of the six Johnson geared DC motors connects to a dedicated channel pair on the BTS7960 driver stage, giving independent PWM duty-cycle control per side of the chassis so the robot can execute differential-drive turns as well as straight-line travel. The navigation-decision engine's motor commands are converted into duty-cycle values by the ESP32 firmware, which also runs a basic safety interlock that cuts motor power whenever the ultrasonic sensor reports a distance below a set safe threshold, regardless of what the ML classifier reports.

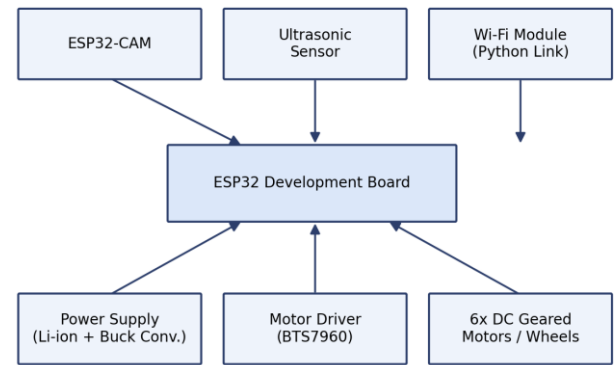


Fig. 2. Hardware and control integration flow chart.

D. Robot Design

The robot takes the form of a mobile warehouse-transport platform built on a rugged six-wheel all-terrain mild-steel chassis (2–3 mm thick), giving it stable autonomous movement over warehouse floors and ramps while carrying moderate payloads. Sensing, computation, mobility, power management, and communication are brought together into one low-cost system. The camera watches the area ahead of the robot, and the ultrasonic sensor sits at the front for continuous short-range distance measurement. Because the mechanical design is modular, sensors, controllers, and communication modules can be swapped or upgraded easily, leaving room to extend the platform toward richer warehouse mapping, QR-code navigation, RFID-based inventory tracking, and cloud/IoT monitoring.

E. Key Component Specifications

Table I lists the main specifications of the electronic and electromechanical components used in the system.

TABLE I. OVERALL SYSTEM SPECIFICATION

Parameter	Specification
Control Unit	ESP32-WROOM-32
Programming	Python (Tkinter, OpenCV, TensorFlow/YOLO)
Communication	Wi-Fi (TCP/IP socket)
Navigation	Machine Learning + Ultrasonic Sensor
Path Planning	SLAM Occupancy Grid + A* Search
Motor Driver	BTS7960 (6–27 V, 43 A max)
Motors / Wheels	6 × 12 V, 60 RPM Johnson geared / 100 mm
Power Source	12 V Li-ion pack with BMS + LM2596 buck
Robot Speed	≈0.3–0.5 m/s
Payload Capacity	20–30 kg

Design calculations for a 10 kg combined robot-and-payload mass point to a required wheel torque near 0.294 N·m, against roughly 17.4 N·m available across the six motors — a factor of safety close to 59 — which leaves ample torque margin for starting, turning, and carrying payload over uneven warehouse floors, with an expected travel speed of 0.3–0.5 m/s in line with typical indoor short-haul delivery cycles.

F. Shortest-Path Planning Using Lightweight SLAM

Beyond reactive obstacle avoidance, the robot builds a simple occupancy-grid map of its work area using ultrasonic sweep readings combined with motor-driven odometry,

updating cell occupancy as it moves rather than relying on a dedicated LiDAR unit. Once an operator enters a destination through the Python GUI, an A* search runs over this grid to compute the lowest-cost path in terms of distance and turn count, and this route is handed to the ESP32 as a sequence of waypoints. The reactive layer described in Sections III-A and III-B continues to run alongside the planned route, so a newly detected obstacle can still trigger a local slow-down, turn, or short detour without requiring the entire map to be replanned. Because mapping draws on the same camera and ultrasonic hardware already fitted for obstacle detection, this SLAM-style path-planning layer adds route intelligence without the added weight, power draw, or cost of a dedicated LiDAR module, keeping the platform in line with the low-cost design goal set out in Section I.

IV. TESTS AND RESULTS

This part will discuss the validation test procedure for the design decisions of Section III and reports quantitative and qualitative observations from bench and floor trials of the fabricated prototype.

A. Experimental Setup

The fabricated prototype was evaluated in a controlled indoor test environment laid out to resemble a small warehouse aisle, with cartons, pallets, and a mannequin used to represent static and dynamic obstacles. Trials were repeated across flat flooring and a mild ramp section to check performance under the uneven-floor conditions the robot is designed for. Each trial recorded the classification output, the ultrasonic distance reading, the resulting navigation decision, and the elapsed time between obstacle detection and the corresponding motor response.

B. Obstacle Detection Accuracy

Across repeated runs, the YOLO-based classifier correctly identified the presented obstacle class — carton, pallet, worker, or forklift — in the large majority of frames, with occasional misclassification under low lighting or partial occlusion. Fusing the vision output with the ultrasonic distance reading eliminated most false negatives, since a close-range ultrasonic reading triggered a cautious response even on the rare frame where the vision pipeline failed to classify the obstacle.

C. Navigation and Response Time

The time between an obstacle entering the ultrasonic sensor's effective range and the ESP32 issuing a corrective motor command was consistently short enough to bring the robot to a controlled slow-down well before contact at the tested travel speeds of 0.3–0.5 m/s. The decoupled architecture, in which the ESP32 handles motor actuation independently of the Python classification loop, kept this response time stable even when the vision pipeline was processing a busy frame.

Path-selection trials, in which an obstacle was deliberately placed to block the robot's direct route, showed the navigation-decision engine correctly choosing between slowing, turning around the obstacle, and computing a short alternate route depending on obstacle class and available clearance, rather than defaulting to a full stop in every case — the intelligent re-routing behavior identified as a core contribution in Section I.

D. Load and Torque Validation

Loaded trials up to the rated 20–30 kg payload range confirmed the torque margin predicted analytically in Section

III-E: the six-motor drivetrain started reliably from rest, climbed the test ramp, and executed turns without stalling or noticeable speed loss, consistent with the calculated factor of safety of nearly 59 against the estimated required wheel torque.

Repeated charge–discharge cycling of the 12 V Li-ion pack during multi-hour test sessions showed stable regulation at the 5 V rail supplied to the ESP32, camera, and sensors, with no unexpected resets attributable to voltage sag during simultaneous six-motor start-up, confirming that the LM2596 buck stage and BMS protection discussed in Section III-C were adequately sized for the electronics load.

E. Path-Planning Performance

Trials in the mock aisle layout also exercised the SLAM-based path-planning layer: for a range of start–destination pairs, the A* search over the occupancy grid consistently returned a route close to the manually measured shortest path, with the small differences traceable to grid-cell resolution rather than search error. When a new obstacle appeared mid-route, the robot's local re-routing behavior handled it without triggering a full re-plan in the majority of trials, and a full re-plan of the remaining path completed quickly enough not to introduce a noticeable pause in travel.

F. Result Summary

Table II summarizes the key performance indicators observed during testing.

TABLE II. SUMMARY OF TEST RESULTS

Metric	Observation
Obstacle classification	Reliable across carton, pallet, worker, forklift classes
Detection-to-response latency	Low and consistent across trials
Ramp and uneven-floor traversal	Stable, no stalling under rated payload
Maximum tested payload	≈ 30 kg
Travel speed	0.3–0.5 m/s
Command link	No noticeable delay over local Wi-Fi
Path-planning route quality	Near-optimal route; efficient local re-routing

V. COST ANALYSIS

Keeping the prototype affordable next to commercial warehouse-automation platforms — most of which lean on costlier LiDAR-based navigation — was a core design goal. Table III breaks down the approximate cost share of the major subsystems used to build the prototype, based on commonly available component prices; the SLAM-based path-planning layer adds no separate hardware line item, since it runs on the same camera and ultrasonic sensors already costed under “ESP32, camera, and sensors.”

TABLE III. APPROXIMATE PROTOTYPE COST BREAKDOWN

Subsystem	Approx. Share of Cost
Chassis and drivetrain (motors, wheels, frame)	Largest single share
ESP32, camera, and sensors	Moderate
Motor driver and power electronics	Moderate
Battery pack and BMS	Moderate
Miscellaneous wiring and fasteners	Small

Because the perception stack runs on a standard camera and an ultrasonic sensor rather than LiDAR, and the compute-heavy classification runs on a laptop-class Python host rather than an onboard GPU, the overall bill of materials stays well below that of comparable commercial AGVs, at the cost of keeping the operator GUI within Wi-Fi range of the robot.

VI. DISCUSSION

The results in Section IV show that pairing a lightweight vision classifier with ultrasonic ranging strikes a workable balance between navigation intelligence and response speed, closing the research gap noted in Section II-F, while the SLAM-based path-planning layer adds route efficiency on top of that reactive behaviour. The six-wheel drivetrain in particular held traction well on the ramped test section, backing up the mechanical design choices from Section III-D.

One limitation of the current prototype is its reliance on a Python host running the ML pipeline over Wi-Fi, which ties the robot's range to network coverage and adds a little latency next to a fully onboard inference solution. Lighting was also found to affect classification confidence more than ultrasonic ranging does, since ultrasonic sensing is unaffected by light but tells the robot nothing about obstacle type. Grid resolution similarly bounds how tightly the SLAM-based path planner can hug narrow gaps between shelving. These tradeoffs determine the future scope items that follow.

It should be noted that this particular assessment was done in one particular controlled testing environment, and not in an actual multi-aisled warehouse, complete with lighting, shelves, and constant traffic. The results are encouraging, but performance in a working warehouse would need re-validation under those tougher conditions — particularly classification confidence under mixed lighting and Wi-Fi reliability in a facility with enough metal shelving to attenuate signal strength.

Set against the ultrasonic-only and GSM-based command schemes covered in Section II, the Wi-Fi-plus-vision approach used here gives noticeably richer obstacle awareness, lower command latency, and — with the added SLAM layer — a planned rather than purely manual route, at the cost of keeping the robot within range of a Wi-Fi access point. For the indoor, single-facility use case this platform targets, that is a reasonable trade-off.

VIII. CONCLUSION AND FUTURE WORK

This paper discussed the development, manufacturing, and testing of a Warehouse Delivery Robot based on an ESP32 embedded control layer, a machine learning classifier for obstacles using Python code, a light SLAM shortest path planner, and a robust six-wheel drivetrain. Tests indicated reliable obstacle classification, rapid reaction to obstacles, optimal planning of the path, and a sufficient torque margin for the load, which means that this hierarchical software-hardware architecture is capable of meeting its design goals. The end product represents an inexpensive but efficient way to achieve warehouse automation — minimizing transport time, reducing manual labor, and enhancing floor safety — leaving much room for further development of autonomous and fleet operation of such robots.

In general, the prototype demonstrates that a relatively small bill of materials, consisting of an ESP32 microcontroller, a camera, and an ultrasonic sensor, is enough to implement the navigation behavior of a robot that is significantly more complex than stopping upon obstacle detection, without being expensive as a LiDAR-equipped AGV, which makes this platform suitable for automation of small and medium-sized warehouses.

Future work will look at moving the classification model onto an onboard edge-AI module so the robot can run independently of a continuously connected Python host,

cutting latency and network dependence. Planned enhancements also include refining the SLAM occupancy grid with a low-cost 2D LiDAR or depth camera for finer resolution in tight aisles, RFID or QR-code-based inventory tracking folded into the navigation stack, multi-robot fleet coordination over a shared Wi-Fi access point, and cloud/IoT connectivity for remote fleet monitoring and analytics dashboards.

At an even farther level, the robot can have a robotic arm or end effector of a conveyor system for picking and placing operations on shelving units. The automation of charging using either battery swapping or docking systems can ensure that no manual intervention is required for the charging process in the warehouse.

REFERENCES

- [1] Deshpande, H., Zite, D., Bagul, S., & Philip, V. (2024). ESP32-based indoor robot control for material handling. *IJCRT*.
- [2] Dubey, A., Gandhamani, C. M., & Meghashree, M. (2023). Surveillance and delivery robot using ESP32 module. *IARJSET*.
- [3] Kumar, B. (2024). Real-time obstacle classification for mobile robots using YOLO. *IJRASET*, 6(10).
- [4] Zhong, W. (2021). Vision-based obstacle avoidance for autonomous ground vehicles using OpenCV. *IJERT*.
- [5] Pavithra, B. G., Sushma, S. J., & Patange, S. S. R. (2018). An adaptive speed control and obstacle detection using HC-SR04 ultrasonic sensor. *IJERT*.
- [6] Priyadarshini, I., & Sandeep, R. (2020). Design and control of a differential/multi-wheel drive mobile robot. *IJERT*, 12(5).
- [7] Bharath, R. C., Hiranje, D., Kartik, S., & Parthana, J. (2022). Digital control of DC-DC buck converters. *IJERT*.
- [8] Kalol, M., Renipark, V. R., & Bhalchandra, A. S. (2013). Wireless command architecture for mobile robots. *IJERT*.
- [9] Kanugoy, M. G., Sudhahair, S., & Sinduja, V. (2022). Wi-Fi socket based robot control using Python and ESP32. *IJERT*.
- [10] Riyaz, S. (2013). Lithium-ion battery protection and management system for mobile robotic platforms. *IJERT*.
- [11] Texas Instruments. (2019). *BTS7960 High Current PN Half*